

UNIT - I

Introduction: AI problems, foundation of AI and history of AI intelligent agents: Agents and Environments, the concept of rationality, the nature of environments, structure of agents, problem solving agents, problem formulation.

Artificial Intelligence - An Introduction

What is AI?

Artificial intelligence is the study of how to make computers do things which, at the moment people do better.

Some definitions of artificial intelligence, organized into four categories

I. Systems that think like humans

1. "The exciting new effort to make computers think machines with minds, in the full and literal sense." (**Haugeland, 1985**)

2. "The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning" (**Bellman, 1978**)

II. Systems that act like humans

3. "The art of creating machines that performs functions that require intelligence when performed by people." (**Kurzweil, 1990**)

4. "The study of how to make computers do things at which, at the moment, people are better." (**Rich and Knight, 1991**)

III. Systems that think rationally

5. "The study of mental faculties through the use of computational models."

(**Chamiak and McDermott, 1985**)

6. "The study of the computations that make it possible to perceive, reason, and act."

(**Winston, 1992**)

IV. Systems that act rationally

7. "Computational Intelligence is the study of the design of intelligent agents."

(**Poole et al., 1998**)

8. "AI is concerned with intelligent behavior in artifacts." (**Nilsson, 1998**)

The definitions on the 1, 2, 3, 4 measure success in terms of human performance, whereas the

rational if it does the "right thing," given what it knows.

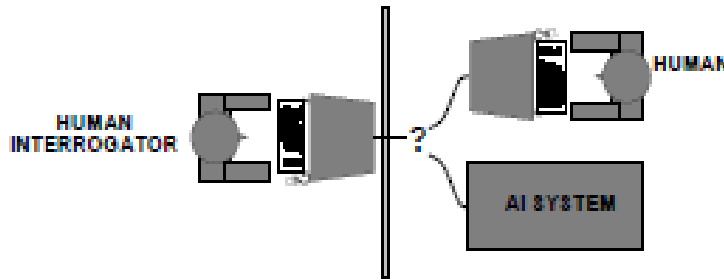
The term AI is defined by each author in its own perceive, leads to four important categories

- i. Acting humanly: The Turing Test approach
- ii. Thinking humanly: The cognitive modeling approach
- iii. Thinking rationally: The "laws of thought" approach

iv. Acting rationally: The rational agent approach

Ones on the 5, 6, 7, 8 measure against an ideal concept of intelligence. A system is Acting humanly: The Turing Test approach

To conduct this test, we need two people and the machine to be evaluated. One person plays the role of the interrogator, who is in a separate room from the computer and the other person. The interrogator can ask questions of either the person or the computer but typing questions and receiving typed responses. However, the interrogator knows them only as A and B and aims to determine which the person is and which is the machine.



The goal of the machine is to fool the interrogator into believing that is the person. If the machine succeeds at this, then we will conclude that the machine is acting humanly. But programming a computer to pass the test provides plenty to work on, to possess the following capabilities.

- ◆ **Natural language processing** to enable it to communicate successfully in English.
- ◆ **Knowledge representation** to store what it knows or hears;
- ◆ **Automated reasoning** to use the stored information to answer

questions and to draw new conclusions

- ◆ **Machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

Total Turing Test: the test which includes a video so that the interrogator can test the perceptual abilities of the machine. To undergo the total Turing test, the computer will need

- ◆ computer vision to perceive objects, and
- ◆ robotics to manipulate objects and move about

i) Thinking humanly: The cognitive modeling approach

To construct a machines program to think like a human, first it requires the knowledge about the actual workings of human mind. After completing the study about human mind it is possible to express the theory as a computer program.

If the program's inputs/output and timing behavior matched with the human behavior then we can say that the program's mechanism is working like a human mind.

Example: General Problem Solver (GPS) - A problem solvers always keeps track of human mind regardless of right answers. The problem solver is contrast to other researchers, because they are concentrating on getting the right answers regardless of the human mind.

An Interdisciplinary field of cognitive science uses computer models from AI and experimental techniques from psychology to construct the theory of the working of the human mind.

ii) Thinking rationally: The "laws of thought" approach

Laws of thought were supposed to govern the operation of the mind and their study initiated the field called **logic**

Example 1:"Socrates is a man; All men are mortal; therefore, Socrates is mortal."

Example 2:"Ram is a student of III year CSE; All students are good in III year CSE;

therefore, Ram is a good student”

Syllogisms : A form of [deductive reasoning](#) consisting of a major [premise](#), a minor premise, and a conclusion

Syllogisms provided patterns for argument structures that always yielded correct conclusions when given correct premises

There are two main obstacles to this approach.

1. It is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less.
2. There is a big difference between being able to solve a problem "in principle" and doing so in practice

iii) Acting rationally: The rational agent approach

An **agent** is just something that acts. A rational **agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome. The study of rational agent has two advantages.

1. Correct inference is selected and applied
2. It concentrates on scientific development rather than other methods.

Foundation of Artificial Intelligence

AI derives the features from Philosophy, Mathematics, Psychology, Computer Engineering, Linguistics topics.

Philosophy(428 B.C. - present)

Aristotle (384-322 B.C.) was the first to formulate a precise set of laws governing the rational part of the mind. He developed an informal system of syllogisms for proper reasoning, which allowed one to generate conclusions mechanically, given initial premises.

Mathematics (c. 800-present)

- ◆ What are the formal rules to draw valid conclusions?
- ◆ What can be computed?
- ◆ How do we reason with uncertain information?

Philosophers staked out most of the important ideas of AI, but the leap to a formal science required a level of mathematical formalization in three fundamental areas: logic, computation, and probability

Economics (1776-present)

- ◆ How should we make decisions so as to maximize payoff?
- ◆ How should we do this when others may not go along?

The science of economics got its start in 1776, when Scottish philosopher Adam Smith (1723- 1790) published An Inquiry into the Nature and Causes of the Wealth of Nations. While the ancient Greeks and others had made contributions to economic thought, Smith was the first to treat it as a science, using the idea that economies can be thought of as consisting of individual agents maximizing their own economic well-being

Neuroscience (1861-present)

- ◆ How do brains process information?

Neuroscience is the study of the nervous system, particularly the brain. The exact way in which the brain enables thought is one of the great mysteries of science. It has been appreciated for thousands of years that the brain is somehow involved in thought, because of the evidence that strong blows to the head can lead to mental incapacitation

	Computer	Human Brain
--	----------	-------------

Computational units	1 CPU, 10^8 gates	10^{11} neurons
Storage units	10^{10} bits RAM	10^{11} neurons
Cycle time	10^{11} bits disk	10^{14} synapses
Bandwidth	10^{-9} sec	10^{-3} sec
Memory updates/sec	10^{10} bits/sec	10^{14} bits/sec
	109	10^{14}
Comparison of the raw computational resources and brain.		

Psychology (1879 - present)

The origin of scientific psychology are traced back to the work of German physiologist Hermann von Helmholtz (1821-1894) and his student Wilhelm Wundt (1832 - 1920). In 1879, Wundt opened the first laboratory of experimental psychology at the University of Leipzig. In US, the development of computer modeling led to the creation of the field of **cognitive science**. The field can be said to have started at the workshop in September 1956 at MIT.

Computer engineering (1940-present)

For artificial intelligence to succeed, we need two things: intelligence and an artifact. The computer has been the artifact of choice. AI also owes a debt to the software side of computer science, which has supplied the operating systems, programming languages, and tools needed to write modern programs.

Control theory and Cybernetics (1948-present)

Ktesibios of Alexandria (c. 250 B.C.) built the first self-controlling machine: a water clock with a regulator that kept the flow of water running through it at a constant, predictable pace. Modern control theory, especially the branch known as stochastic optimal control, has as its goal the design of systems that maximize an **objective function** over time.

Linguistics (1957-present)

Modern linguistics and AI, then, were "born" at about the same time, and grew up together, intersecting in a hybrid field called **computational linguistics** or **natural language processing**.

History of Artificial Intelligence

The gestation of artificial intelligence (1943-1955)

There were a number of early examples of work that can be characterized as AI, but it was Alan Turing who first articulated a complete vision of AI in his 1950 article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.

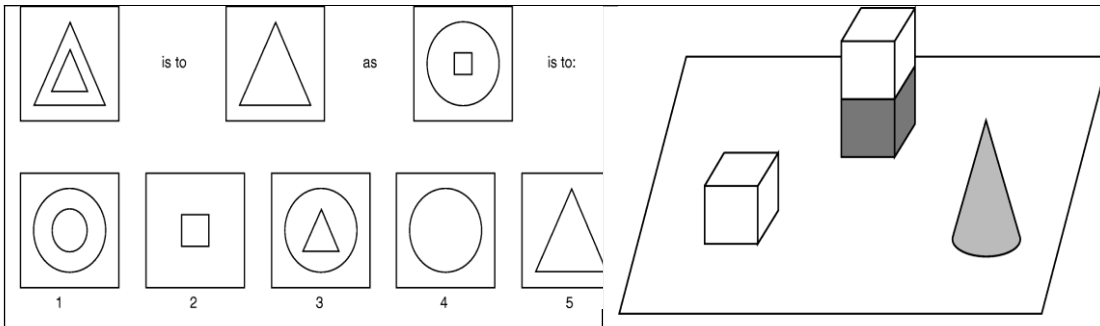
The birth of artificial intelligence (1956)

McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help him bring together U.S. researchers interested in automata theory, neural nets, and the study of intelligence. They organized a two-month workshop at Dartmouth in the summer of 1956. Perhaps the longest-lasting thing to come out of the workshop was an agreement to adopt McCarthy's new name for the field: **artificial intelligence**.

Early enthusiasm, great expectations (1952-1969)

The early years of AI were full of successes in a limited way. **General Problem Solver (GPS)** was a computer program created in 1957 by Herbert Simon and Allen Newell to build a universal problem solver machine. The order in which the program considered subgoals and possible actions was similar to that in which humans approached the same problems. Thus, GPS was probably the first program to embody the "thinking humanly" approach. At IBM, Nathaniel Rochester and his colleagues produced some of the first AI programs. Herbert Gelernter (1959) constructed the Geometry Theorem Prover, which

was able to prove theorems that many students of mathematics would find quite tricky. Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute of Technology (MIT). In 1963, McCarthy started the AI lab at Stanford. Tom Evans's ANALOGY program (1968) solved geometric analogy problems that appear in IQ tests, such as the one in Figure



A dose of reality (1966-1973)

From the beginning, AI researchers were not shy about making predictions of their coming successes. The following statement by Herbert Simon in 1957 is often quoted:

"It is not my aim to surprise or shock you-but the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until-in a visible future-the range of problems they can handle will be coextensive with the range to which the human mind has been applied.

Knowledge-based systems: The key to power? (1969-1979)

Dendral was an influential pioneer project in artificial intelligence (AI) of the 1960s, and the computer software **expert system** that it produced. Its primary aim was to help organic chemists in identifying unknown organic molecules, by analyzing their mass spectra and using knowledge of chemistry. It was done at Stanford University by Edward Feigenbaum, Bruce Buchanan, Joshua Lederberg, and Carl Djerassi.

AI becomes an industry (1980-present)

In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build intelligent computers running Prolog. Overall, the AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988.

The return of neural networks (1986-present)

Psychologists including David Rumelhart and Geoff Hinton continued the study of neural-net models of memory.

AI becomes a science (1987-present)

In recent years, approaches based on **hidden Markov models** (HMMs) have come to dominate the area. Speech technology and the related field of handwritten character recognition are already making the transition to widespread industrial and consumer applications.

The **Bayesian network** formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge.

The emergence of intelligent agents (1995-present)

One of the most important environments for intelligent agents is the Internet.

Sample Applications

Autonomous planning and scheduling: A hundred million miles from Earth, NASA's Remote Agent program became the first on-board autonomous planning program to control the scheduling of operations for a spacecraft. Remote Agent generated plans from high-level goals specified from the ground, and it monitored the operation of the spacecraft as the plans were executed-detecting, diagnosing, and recovering from problems as they occurred.

Game playing: IBM's Deep Blue became the first computer program to defeat the world champion (Garry Kasparov) in a chess match. The value of IBM's stock increased by \$18

billion.

Autonomous control: The ALVINN computer vision system was trained to steer a car to keep it following a lane. The computer-controlled minivan used to navigate across the United States- for 2850 miles and it was in control of steering the vehicle 98% of the time. A human took over the other 2%, mostly at exit ramps.

Diagnosis: Medical diagnosis programs based on probabilistic analysis have been able to perform at the level of an expert physician in several areas of medicine

Logistics Planning: During the Gulf crisis of 1991, U.S. forces deployed a Dynamic Analysis and Replanning Tool, DART to do automated logistics planning and scheduling for transportation. This involved up to 50,000 vehicles, cargo, and people at a time, and had to account for starting points, destinations, routes, and conflict resolution

Robotics: Many surgeons now use robot assistants in microsurgery.

Language understanding and problem solving: PROVERB is a computer program that solves crossword puzzles better than most humans, using constraints on possible word fillers, a large database of past puzzles, and a variety of information sources including dictionaries and online databases such as a list of movies and the actors that appear in them.

Typical problems to which AI methods are applied

[Pattern recognition](#), [Optical character recognition](#), [Handwriting recognition](#), [Speech recognition](#), [Face recognition](#), [Computer vision](#), [Virtual reality](#) and [Image processing](#), [Diagnosis](#), [Game theory](#) and [Strategic planning](#), [Natural language processing](#), [Translation](#) and [Chatterboxes](#), [Nonlinear control](#) and [Robotics](#), [Artificial life](#), [Automated reasoning](#), [Automation](#), [Biologically inspired computing](#), [Concept mining](#), [Data mining](#), [Knowledge representation](#), [Semantic Web](#), [E-mail spam filtering](#), [Robotics](#), [Cognitive](#), [Cybernetics](#), [Hybrid intelligent system](#), [Intelligent agent](#), [Intelligent control](#)

INTELLIGENT AGENTS

Introduction - Agents and Environments

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

Different types of agents

1. A **human agent** has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators.
2. A **robotic agent** might have cameras and infrared range finders for sensors and various motors for actuators.
3. A **software agent** receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.
4. **Generic agent** - A general structure of an agent who interacts with the environment.

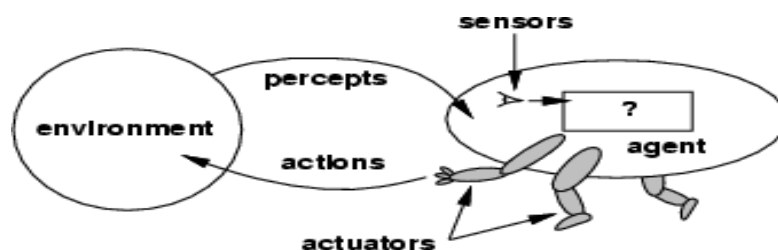


Fig : Agents interact with environments through sensors and effectors (actuators) **Percept** : The term **percept** is to refer to the agent's perceptual inputs at any given instant.

PERCEPT SEQUENCE: Agent's percept sequence is the complete history of everything the agent has ever perceived.

An agent's behavior is described by the agent function that maps any given percept sequence to an action.

AGENT PROGRAM : The agent function for an artificial agent will be implemented by an agent program.

Example : The vacuum-cleaner world has just two locations: squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, suck up the dirt, or do nothing. One very simple agent function is the following: if the current square is dirty, then suck, otherwise move to the other square.

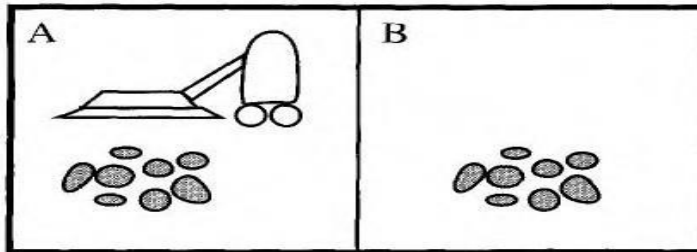


Fig : A vacuum-cleaner world with just two locations

Partial tabulation of a simple agent function for the vacuum-cleaner world

- Percepts: location and status, e.g., [A,Dirty]
- Actions: Left, Right, Suck, NoOp

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck

The agent program for a simple agent in the two-state vacuum environment for above tabulation

```
function VACUUM-AGENT([location,status])
if status = Dirty then return Suck
else if location = A then return
Right else if location = B then
```

Concept of Rationality

A rational agent is one that does the right thing. The right action is the one that will cause the agent to be most successful.

Performance measures

A performance measure embodies the criterion for success of an agent's behavior. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well.

Rationality

Rational at any given time depends on four things:

1. The performance measure that defines the criterion of success.
2. The agent's prior knowledge of the environment.
3. The actions that the agent can perform.
4. The agent's percept sequence to date.

Definition of a rational agent:

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has. A rational agent should be autonomous

Definition of an omniscient agent:

An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.

Autonomy

A rational agent should be autonomous-it should learn what it can to compensate for partial or incorrect prior knowledge.

Information Gathering

Doing actions in order to modify future percepts is called as information gathering.

Specifying the task environment

In the discussion of the rationality of any agent, we had to specify the performance measure, the environment, and the agent's actuators and sensors. We group all these together under the heading of the task environment and we call this as **PEAS (Performance, Environment, Actuators, Sensors)** or **PAGE (Percept, Action, Goal, Environment)** description. In designing an agent, the first step must always be to specify the task environment.

Example : PEAS description of the task environment for agent

Agent Type	Performance Measure	Environment	Actuators	Sensors
Automated Taxi Driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, traffic, pedestrian customers	Steering accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer engine

				sensors, keyboard
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Screen display (question tests, diagnoses treatment referrals)	Keyboard (entry of symptoms, findings, patient's answers)
Part-Picking Robot	Percentage of parts in correct Bin	Conveyor belt with parts, bins	Jointed arm and hand	Camera, joint angle sensors
Interactive English tutor	Maximize student's score on test	Set of students	Screen display (exercises)	Keyboard
robot soccer player	amount of goals scored	soccer match field	legs	cameras, sonar or infrared
Satellite Image Analysis	Correct Image Categorization	Downlink from satellite	Display categorization of scene	Color pixel arrays
Refinery controller	Maximum purity, safety	Refinery operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Vacuum Agent	minimize energy consumption, maximize dirt pick up	two squares	Left, Right, Suck, NoOp	Sensors to identify the dirt

PROPERTIES OF TASK ENVIRONMENTS (ENVIRONMENT TYPES)

Fully observable vs. partially observable

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable. A chess playing system is an example of a system that operates in a fully observable environment.

An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data. A bridge playing program is an example of a system operating in a partially observable environment.

Deterministic vs. stochastic

If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic

Image analysis systems are examples of deterministic. The processed image is determined completely by the current image and the processing operations.

Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of

traffic exactly;

Episodic vs. sequential

An episodic environment means that subsequent episodes do not depend on what actions occurred in previous episodes.

In a sequential environment, the agent engages in a series of connected episodes. In sequential environments, on the other hand, the current decision could affect all future decisions. Chess and taxi driving are sequential

Static vs. dynamic

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static. Taxi driving is clearly dynamic. Crossword puzzles are static.

Discrete vs. continuous

If the number of distinct percepts and actions is limited, the environment is discrete, otherwise it is continuous. Taxi driving is a continuous state and continuous-time problem. Chess game has a finite number of distinct states.

Single agent vs. Multi agent

The distinction between single-agent and multi agent environments may seem simple enough. For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment.

Chess is a competitive multi agent environment. Taxi-driving environment is a partially cooperative multi agent environment.

Environment Characteristics

Examples of task environments and their characteristics

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agent
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Stochastic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Image- analysis	Fully	Deterministic	Episodic	Semi	Continuous	Single
Part-picking robot	Partially	Stochastic	Episodic	Dynamic	Continuous	Single
Refinery controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi

- The simplest environment is
 - Fully observable, deterministic, episodic, static, discrete and single- agent.
- Most real situations are:
 - Partially observable, stochastic, sequential, dynamic, continuous and multi- agent.

Structure of the Agents

The job of AI is to design the agent program that implements the agent function mapping percepts to actions.

Intelligent agent = Architecture + Agent program

Agent programs

Agent programs take the current percept as input from the sensors and return an action to the actuators

The agent program takes the current percept as input, and the agent function takes the entire percept history

Architecture is a computing device used to run the agent program.

The agent programs will use some internal data structures that will be updated as new percepts arrive. The data structures are operated by the agents decision making procedures to generated an action choice, which is then passed to the architecture to be executed. Two types of agent programs are

1. A Skeleton Agent

2. A Table Lookup Agent Skeleton Agent

The agent program receives only a single percept as its input.

If the percept is a new input then the agent updates the memory with the new percept

```
function SKELETON-AGENT( percept) returns action
static: memory, the agent's memory of the world
memory <- UPDATE-MEMORY(memory, percept)
action <- CHOOSE-BEST-ACTION(memory)
memory <- UPDATE-MEMORY(memory, action)
return action
```

Table-lookup agent

A table which consists of indexed percept sequences with its corresponding action The input percept checks the table for the same

```
function TABLE-DRIVEN-AGENT(percept) returns an action

static: percepts, a sequence initially empty
       table, a table of actions, indexed by percept
       sequence
append percept to the end of percepts
```

Drawbacks of table lookup agent

- Huge table
- Take a long time to build the table
- No autonomy
- Even with learning, need a long time to learn the table entries

Four basic types in order of increasing generality

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

Simple reflex agents

The simplest kind of agent is the simple reflex agent. These agents select actions on the basis of the current percept, ignoring the rest of the percept history.

This agent describes about how the condition - action rules allow the agent to make the connection from percept to action

It acts according to a rule whose condition matches the current state, as defined by the percept.

Condition - action rule : if condition then action

Example : condition-action rule: if car-in-front-is-braking then initiate- braking

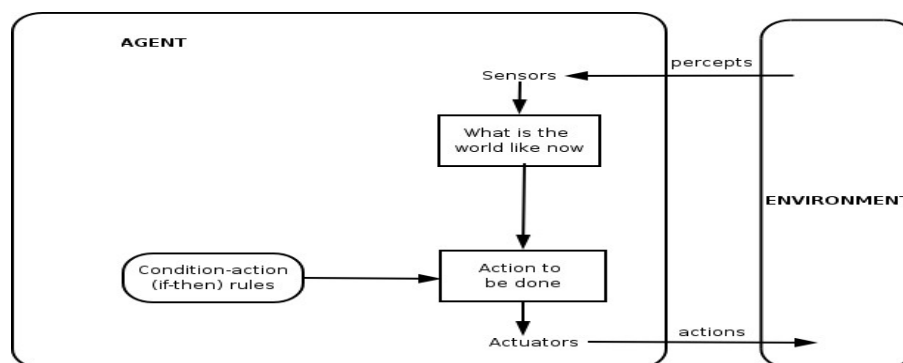


Fig : Schematic diagram of a simple reflex agent.

Rectangles - to denote the current internal state of the agent's decision process

Ovals - to represent the background information used in the process.

```
function SIMPLE-REFLEX-AGENT(percept) returns action
static : rules, a set of condition-action rules
state <- INTERPRET-INPUT(percept)
rule <- RULE-MATCH(state, rules),
action <- RULE-ACTION[rule]
return action
```

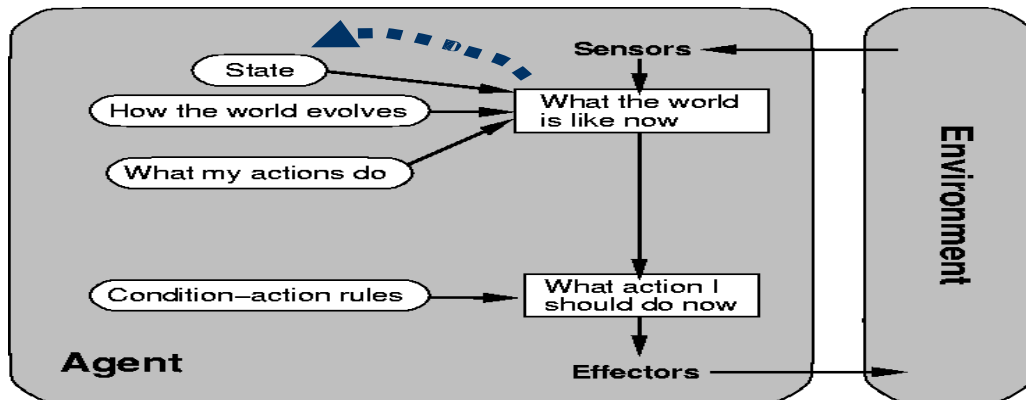
- **INTERPRET-INPUT** function generates an abstracted description of the current state from the percept
- **RULE-MATCH** function returns the first rule in the set of rules that matches the given state description
- **RULE - ACTION** - the selected rule is executed as action of the given percept

Example : Medical Diagnosis System

If the patient has reddish brown spots then start the treatment for measles.

Model based Reflex Agents

An agent which combines the current percept with the old internal state to generate updated description of the current state.



```
function REFLEX-AGENT-WITH-STATE(percept) returns action
static: state, a description of the current world state
       rules, a set of condition-action rules
       action, the most recent action, initially none
state <- UPDATE-STATE( state, action,
percept) rule <- RULE - MATCH ( state, rules
)
```

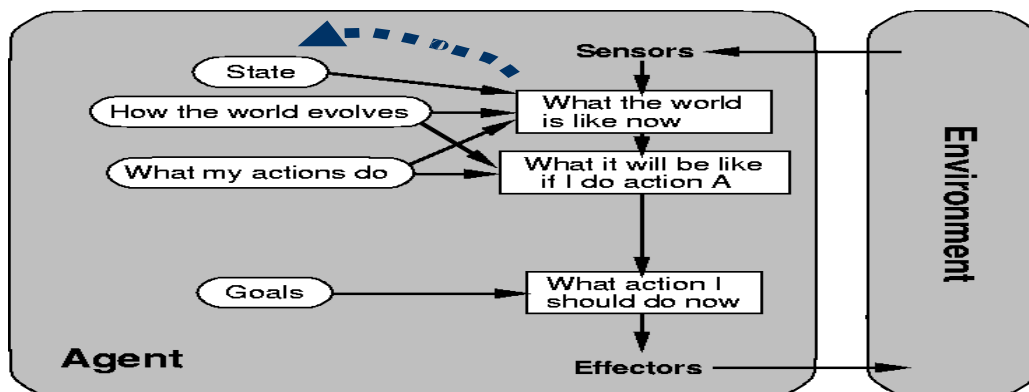
STATE - is responsible for creating the new internal state description

Example: Medical Diagnosis system UPD ATE-

If the Patient has spots then check the internal state (i. e) any change in the environment may lead to cause spots on the patient. From this internal state the current state is updated and the corresponding action is executed.

Goal based Agents

An Agent knows the description of current state as well as goal state. The action matches with the current state is selected depends on the goal state.



Example : Medical diagnosis system

If the name of disease is identified for the patient then the treatment is given to the

patient to recover from him from the disease and make the patient healthy is the goal to be achieved

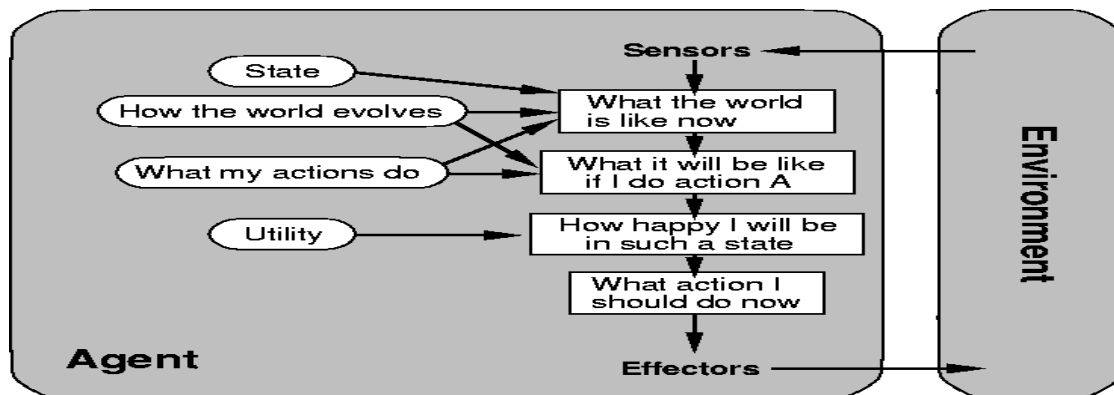
Utility base agents

An agent which generates a goal state with high - quality behavior (i.e) if more than one sequence exists to reach the goal state then the sequence with more reliable, safer, quicker and cheaper than others to be selected.

Utility is a function that maps a state onto a real number, which describes the associated degree of happiness

The utility function can be used for two different cases :

1. When there are conflicting goals, only some of which can be achieved (for example, speed and safety)
2. When there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed up against the importance of the goal



Example : Medical diagnosis System

If the patient disease is identified then the sequence of treatment which leads to recover the patient with all utility measure is selected and applied

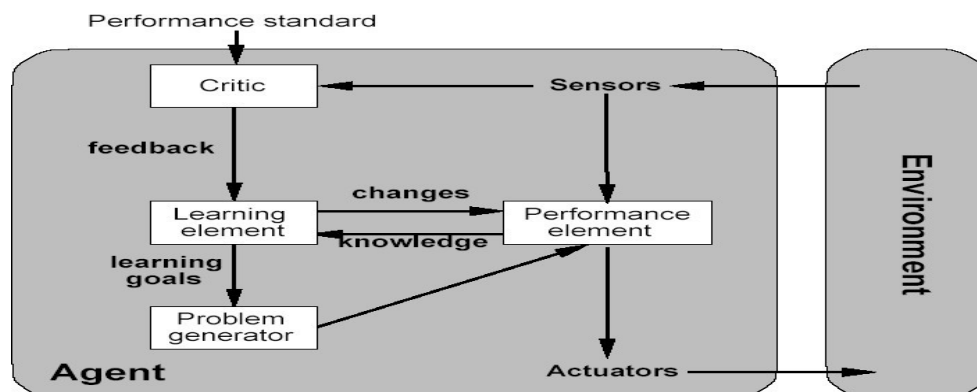
Learning agent

All agents can improve their performance through Learning

The learning task allows the agent to operate in unknown environments initially and then become more competent than its initial knowledge.

A learning agent can be divided into four conceptual components:

1. Learning element
2. performance element
3. Critic
4. Problem generator



The **learning element** uses feedback from the critic on how the agent is doing and

determines how the performance element should be modified to do better in the future. Learning element is also responsible for making improvements

Performance element is to select external action and it is equivalent to agent

The **critic** tells the learning element how well the agent is doing with respect to a fixed performance standard

The last component of the learning agent is the **problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences.

Problem solving - Introduction

Search is one of the operational tasks that characterize AI programs best. Almost every AI program depends on a search procedure to perform its prescribed functions. Problems are typically defined in terms of state, and solution corresponds to goal states.

Problem solving using search technique performs two sequence of steps:

- (i) **Define the problem** - Given problem is identified with its required initial and goal state.
- (ii) **Analyze the problem** - The best search technique for the given: problem is chosen from different an AI search technique which derives one or more goal state in minimum number of states.

Types of problem

In general the problem can be classified under anyone of the following four types which depends on two important properties. They are

- (i) Amount of knowledge, of the agent on the state and action description.
- (ii) How the agent is connected to its environment through its percepts and actions? The four different types of problems are:
 - (i) Single state problem
 - (ii) Multiple state problem
 - (iii) Contingency problem
 - (iv) Exploration problem

Problem solving Agents

Problem solving agent is one kind of goal based agent, where the agent decides what to do by finding sequence of actions that lead to desirable states. The complexity arises here is the knowledge about the formulation process, (from current state to outcome action) of the agent.

If the agent understood the definition of problem, it is relatively straight forward to construct a search process for finding solutions, which implies that problem solving agent should be an intelligent agent to maximize the performance measure.

The sequence of steps done by the intelligent agent to maximize the performance measure:

- i) **Goal formulation** - based on current situation is the first step in problem solving. Actions that result to a failure case can be rejected without further consideration.
- (ii) **Problem formulation** - is the process of deciding what actions and states to consider and follows goal formulation.
- (iii) **Search** - is the process of finding different possible sequence of actions that lead to state of known value, and choosing the best one from the states.
- (iv) **Solution** - a search algorithm takes a problem as input and returns a solution in the form of action sequence.
- (v) **Execution phase** - if the solution exists, the action it recommends can be carried out.

```

function    SIMPLE-PROBLEM-SOLVING-AGENT(p)    returns    an
action
input : p, a percept
static: s, an action sequence, initially empty
state, some description of the current world state
g, a goal initially null
problem, a problem formulation state <- UPDATE-
STATE(state, p) if s is empty then
g <- FORMULATE-GOAL(state)
problem <- FORMULATE-PROBLEM(state, g)
s <- SEARCH(problem)
action <- RECOMMENDATION(s, state) s <-
REMAINDER(s, state)
return action

```

A simple problem solving agent

Note :

RECOMMENDATION - first action in the sequence

REMAINDER - returns the rest

SEARCH - choosing the best one from the sequence of actions **FORMULATE-PROBLEM**
- sequence of actions and states that lead to goal state.

UPDATE-STATE - initial state is forced to next state to reach the goal state

Well-defined problems and solutions

A problem **can be defined formally by four components:**

- 1. initial state**
- 2. successor function**
- 3. goal test**
- 4. path cost**

The **initial state** that the agent starts in.

Successor function (S) - Given a particular state *x*, *S*(*x*) returns a set of states reachable from *x* by any single action.

The **goal test**, which determines whether a given state is a goal state. Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state is one of them.

A **path cost** function that assigns a numeric cost to each path. The problem- solving agent chooses a cost function that reflects its own performance measure.

A **solution** to a problem is a path from the initial state to a goal state

Operator - The set of possible actions available to the agent.

State space (or) state set space - The set of all possible states reachable from the initial state by any sequence of actions.

Path (state space) - The sequence of action leading from one state to another The effectiveness of a search can be measured using three factors. They are:

1 Solution is identified or not?

2. Is it a good solution? If yes, then path cost to be minimum.

3. Search cost of the problem that is associated with time and memory required to find a solution.

For Example

Imagine an agent in the city of Arad, Romania, enjoying a touring holiday. Now, suppose the agent has a nonrefundable ticket to fly out of Bucharest the following day. In that case, it makes sense for the agent to adopt the goal of getting to Bucharest. The agent's task is to find out which sequence of actions will get it to a goal state.

This process of looking for such a sequence is called search.

A search algorithm takes a problem as input and returns a solution in the form of an action sequence. Once a solution is found, the actions it recommends can be carried out. This is called the execution phase.

Formulating problems

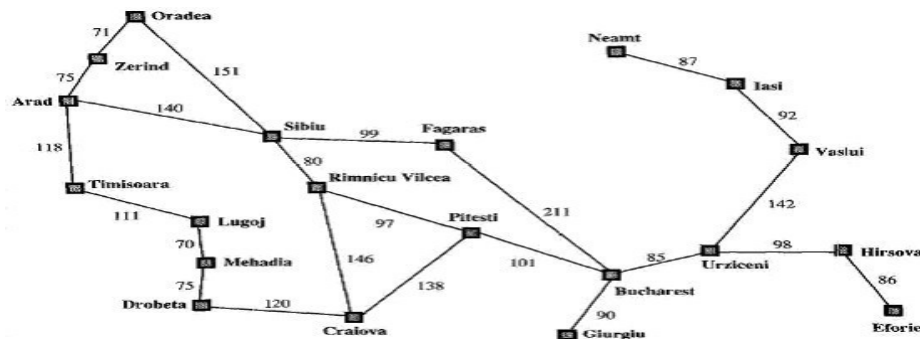
Initial state : the initial state for our agent in Romania might be described as In(Arad)

Successor function : Given a particular state x , $SUCCESSOR-FN(x)$ returns a set of (action, successor) ordered pairs, where each action is one of the legal actions in state x and each successor is a state that can be reached from x by applying the action. For example, from the state In(Arad), the successor function for the Romania problem would return

$\{(Go(Sibiu), In(Sibiu)), (Go(Timisoara), In(Timisoara)), (Go(Zerind), In(Zerind))\}$

Goal test : The agent's goal in Romania is the singleton set $\{In(Bucharest)\}$.

Path cost : The **step cost** of taking action a to go from state x to state y is denoted by $c(x, a, y)$.



Example Problems

The problem-solving approach has been applied to a vast array of task environments.

A toy problem is intended to illustrate or exercise various problem-solving methods. It can be given a concise, exact description. It can be used easily by different researchers to compare the performance of algorithms

A real-world problem is one whose solutions people actually care about. Some list of best known toy and real-world problems

Toy Problems

i) Vacuum world Problem

States: The agent is in one of two locations, each of which might or might not contain dirt. Thus there are $2 * 2^2 = 8$ possible world states.

Initial state: Any state can be designated as the initial state. **Successor function:** three actions (Left, **Right**, and **Suck**). **Goal test:** This checks whether all the squares are clean.

Path cost: Each step costs 1, so the path cost is the number of steps in the path.

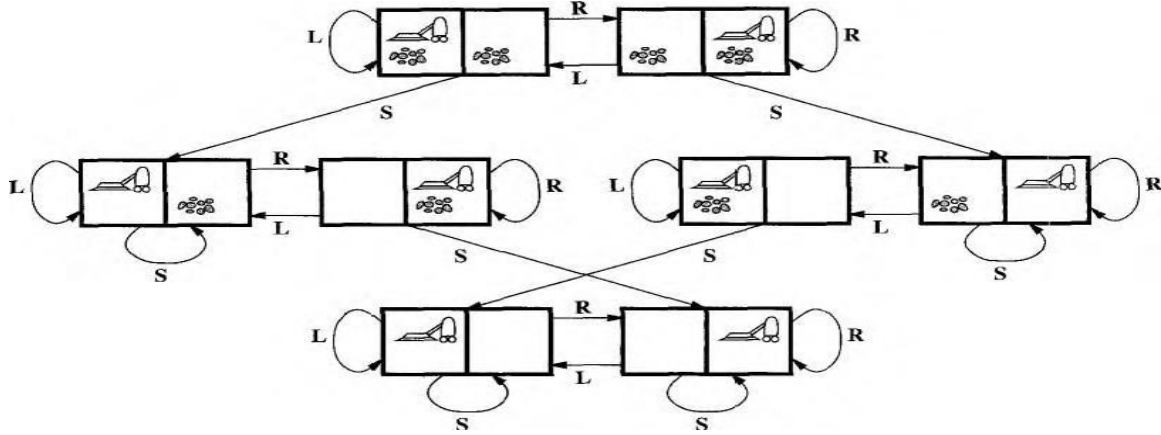


Fig : The complete state space for Vacuum World

ii) 8-puzzle Problem

The 8-puzzle problem consists of a 3 x 3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach a specified goal state

States: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.

Initial state: Any state can be designated as the initial state.

Successor function: This generates the legal states that result from trying the four actions (blank moves Left, Right, Up, or Down).

Goal test: This checks whether the state matches the goal configuration (Other goal configurations are possible.)

Path cost: Each step costs 1, so the path cost is the number of steps in the path.

2	8	3
1	6	4
7		5

	1	2
3	4	5
6	7	8

Initial State

Goal State

iii) 8-queens problem

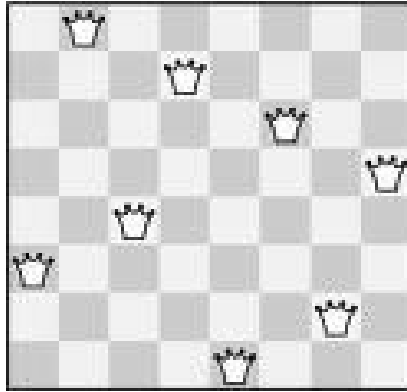
The goal of the **8-queens problem** is to place eight queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.

States: Any arrangement of 0 to 8 queens on the board is a state.

Initial state: No queens on the board.

Successor function: Add a queen to any empty square.

Goal test: 8 queens are on the board, none attacked. Path cost : Zero (search cost only exists)



solution to the 8-queens problem.

iv) Crypt arithmetic Problem

In crypt arithmetic problems letters stand for digits and the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, each letter stand for a different digit

Rules

There should be no more than 10 distinct characters The summation should be the longest word

The summation can not be too long

There must be a one-to-one mapping between letters and digits The leftmost letter can't be zero in any word.

States: A crypt arithmetic puzzle with some letters replaced by digits

Initial state: No digits is assigned to the letters

Successor function: Replace all occurrences of a letter with a digit not already appearing in the puzzle

Goal test: Puzzle contains only digits and represents a correct sum

Path cost : Zero

Example 1:

$$\begin{array}{r}
 \text{S E N D} \\
 + \text{M O R E} \\
 \hline
 \text{M O N E Y}
 \end{array}$$

Solution : S=9 , E = 5, N = 6, D=7, M= 1, O= 0, R = 8, Y=2

Example 2:

$$\begin{array}{r}
 \text{FORTY} \\
 + \text{TEN}
 \end{array}$$

+TEN

SIXTY

Solution : F=2, O=9, R=7, T=8 , Y=6, E=5, N=0

v) Missionaries and cannibals problem

Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place
Assumptions :

1. Number of trips is not restricted
2. Both the missionary and cannibal can row the boat

States: A state consists of an ordered sequence of two numbers representing the number of missionaries and cannibals

Example : $(i,j) = (3,3)$ three missionaries and three cannibals

Initial state: $(i,j) = (3,3)$ in one side of the river

Successor function: The possible moves across the river are:

1. One Missionary and One Cannibal
2. Two Missionaries
3. Two Cannibals
4. One Missionary
5. One Cannibal

Rule No.	Explanation
(i)	(i, j) : One missionary and one cannibal can cross the river only when $((i-1) \geq (j-1))$ in one side of the river and $((i+1) \geq (j+1))$ in the other side of the river.
(ii)	(i,j) : Two missionaries can cross the river only when $((i-2) \geq j)$ in one side of the river and $((i+2) \geq j)$ in the other side of the river.
(iii)	(i,j) : Two cannibals can cross the river only when $((j-2) \leq i)$ in one side of the river and $((j+2) \leq i)$ in the other side of the river.
(iv)	(i,j) : One missionary can cross the river only when $((i-1) \geq j)$ in one side of the river and $((i-1) \geq j)$ in the other side of the river.

Initial state : $(i,j) = (3,3)$ in one side of the river.

Goal test: $(i,j) = (3,3)$ in the other side of the river.

Path cost : Number of crossings between the two sides of the river.

Solution:

Bank1		Boat		Bank2	Rule Applied
(i,j)=(3,3)				(i,j)=(0,0)	
->(3,1)		(0,2)	->	(0,2)	(iii)
<-(3,2)		(0,1)	<-	(0,1)	(v)
(3,0)	->	(0,2)	->	(0,3)	(iii)
(3,1)	<-	(0,1)	<-	(0,2)	(v)
(1,1)	->	(2,0)	->	(2,2)	(ii)
(2,2)	<-	(1,1)	<-	(1,1)	(i)
(0,2)	->	(2,0)	->	(3,1)	(ii)
(0,3)	<-	(0,1)	<-	(3,0)	(v)
(0,1)	->	(0,2)	->	(3,2)	(iii)
(0,2)	<-	(0,1)	<-	(3,1)	(v)
(0,0)	->	(0,2)	->	(3,3)	(iii)

Real-world problems Airline travel problem

States: Each is represented by a location (e.g., an airport) and the current time. Initial state: This is specified by the problem.

Successor function: This returns the states resulting from taking any scheduled flight (perhaps further specified by seat class and location), leaving later than the current time plus the within- airport transit time, from the current airport to another.

Goal test: Are we at the destination by some pre specified time?

Path cost: This depends on monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent- flyer mileage awards, and so on.

Route-finding problem is defined in terms of specified locations and transitions along links between them. Route-finding algorithms are used in a variety of applications, such as routing in computer networks, military operations planning, and airline travel planning systems

The **traveling salesperson problem** (TSP) is a touring problem in which each city must be visited exactly once. The aim is to find the shortest tour.

A VLSI **layout** problem requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield. The layout problem comes after the logical design phase, and is usually split into two parts: **cell layout** and **channel routing**. In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognized function. Each cell has a fixed footprint (size and shape) and requires a certain number of connections to each of the other cells. The aim is to place the cells on the chip so that they do not overlap and so that there is room for the connecting wires to be placed between the cells. Channel routing finds a specific route for each wire through the gaps between the cells.

Robot navigation is a generalization of the route-finding problem described earlier. Rather than a discrete set of routes, a robot can move in a continuous space with (in principle) an infinite set of possible actions and states. For a circular robot moving on a flat surface, the space is essentially two-dimensional. When the robot has arms and legs or wheels that must also be controlled, the search space becomes many-dimensional. Advanced techniques are required just to make the search space finite. In addition to the

complexity of the problem, real robots must also deal with errors in their sensor readings and motor controls.

Automatic assembly sequencing of complex objects by a robot was first demonstrated by FREDDY (Michie, 1972). In assembly problems, the aim is to find an order in which to assemble the parts of some object. **If** the wrong order is chosen, there will be no way to add some part later in the sequence without undoing some of the work already done. Checking a step in the sequence for feasibility is a difficult geometrical search problem closely related to robot navigation.

